

CSCI 3753: Operating Systems

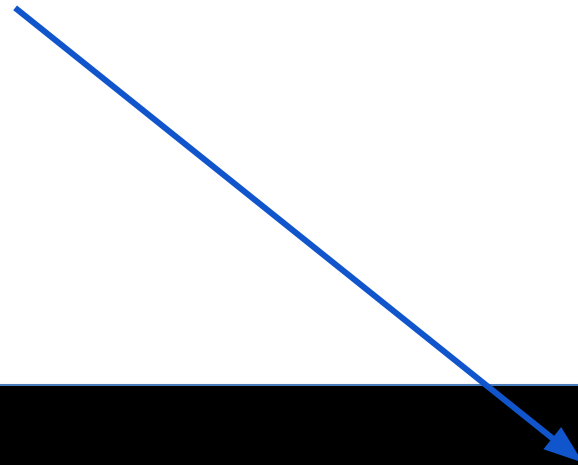
Week 3
September 12, 2025





University of Colorado **Boulder**

Announcements

- PA0+1 Interview Grading almost over!
- Quiz 3 due at midnight
- Problem Set 1 due on Sunday
- PA2 and PA3 are SIGNIFICANTLY MORE WORK THAN PA0 and PA1
- Recitation materials: <https://tinyurl.com/CSCI3753>

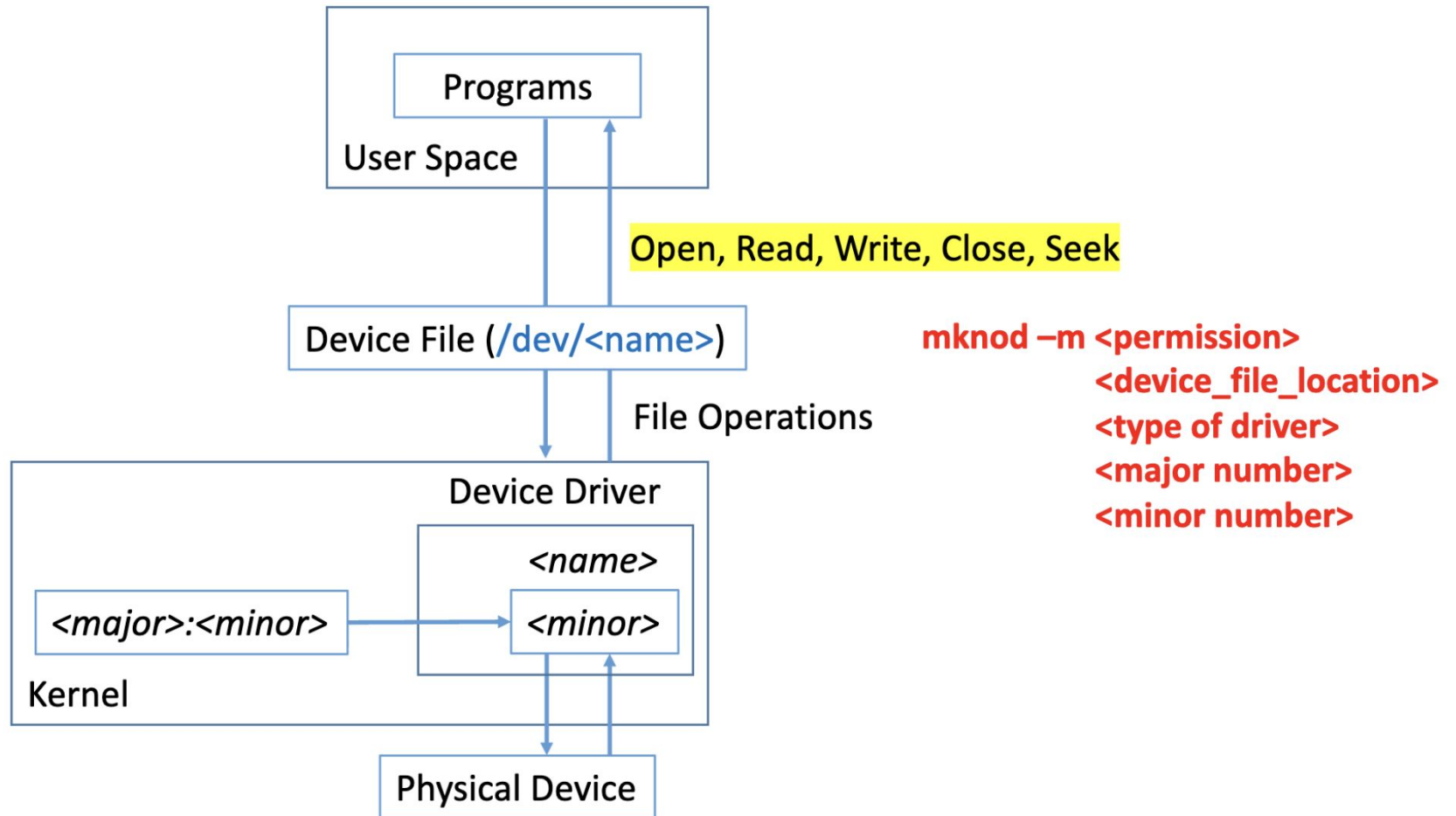


Recap

- PA0 
- PA1 
- User/Kernel Space
- System Calls
- LKM



PA 3



PA 3

- create a Device Driver Module (LKM)
- implement file operations
 - open, seek, read, write, release
- make and load the module
- create a Device File for this Device



PA 3

- create a Device Driver Module (LKM)
 - implement file operations
 - open, seek, read, write, release
 - make and load the module
 - create a Device File for this Device
-
- **create a test program**

PA 2



PA 2

- Test program for PA 3
- pa2test.c
 - infinite loop with the following features
 - r - read()
 - w - write()
 - s - seek()
 - SEEK_SET
 - SEEK_CUR
 - SEEK_END
 - control+d for termination
 - other entries should be ignored



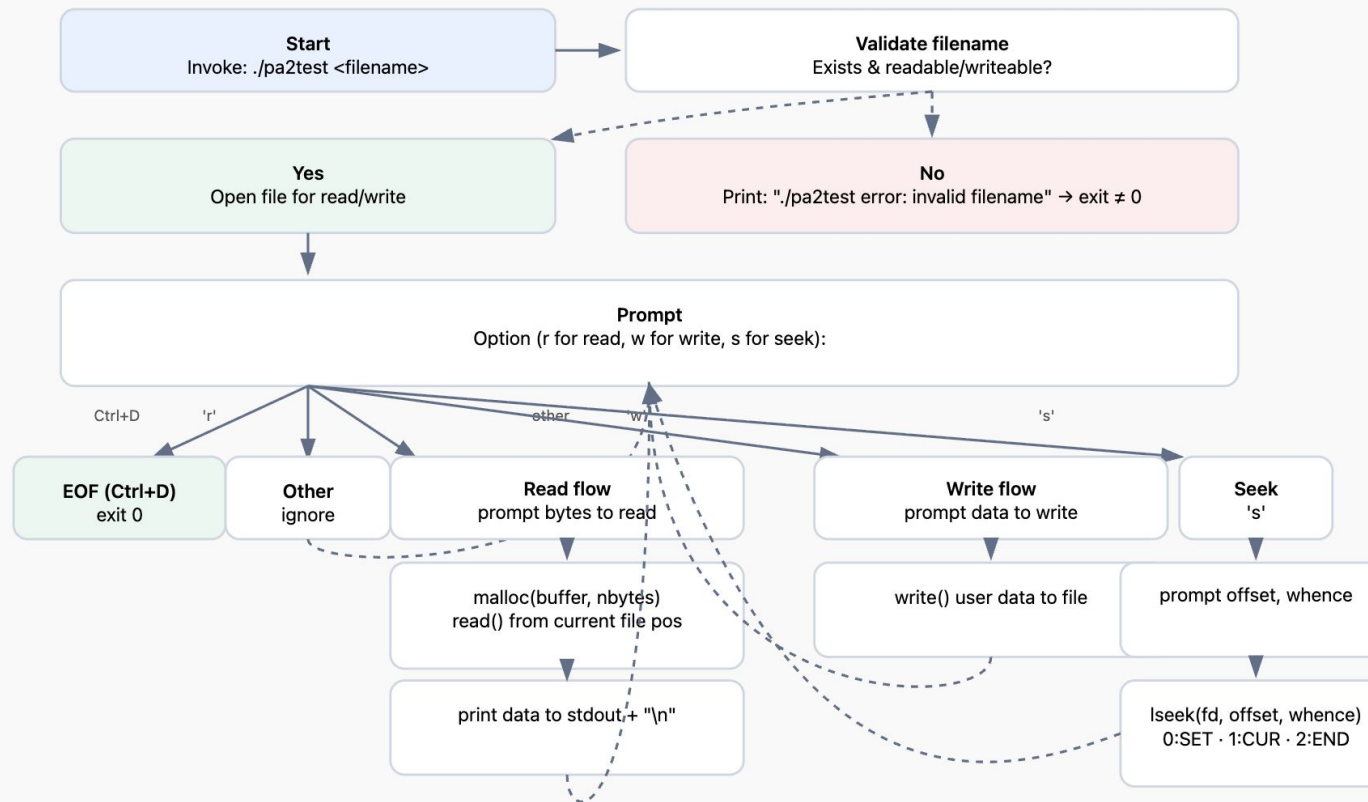
PA 2 Flow

Exit codes: 0 on normal (Ctrl+D) termination · ≠0 on error (e.g., invalid filename).

Seek *whence*: 0 → SEEK_SET , 1 → SEEK_CUR , 2 → SEEK_END .

Loop behavior: unknown/other input is ignored; prompt repeats.

Reads use a `malloc()` 'd buffer sized to requested bytes.



PA2 Demo

- Available on Canvas

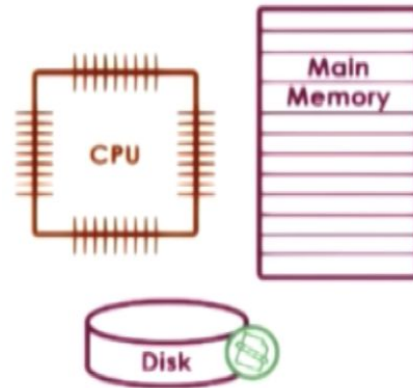


Processes and Threads

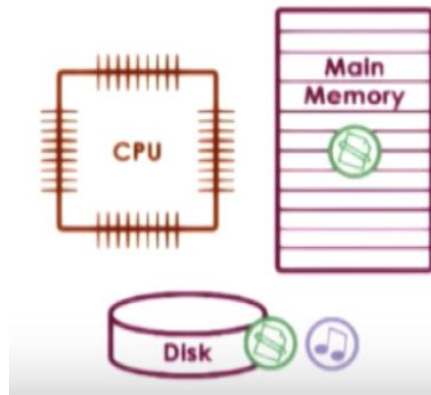


What is a Process?

OS manages hardware on behalf of applications



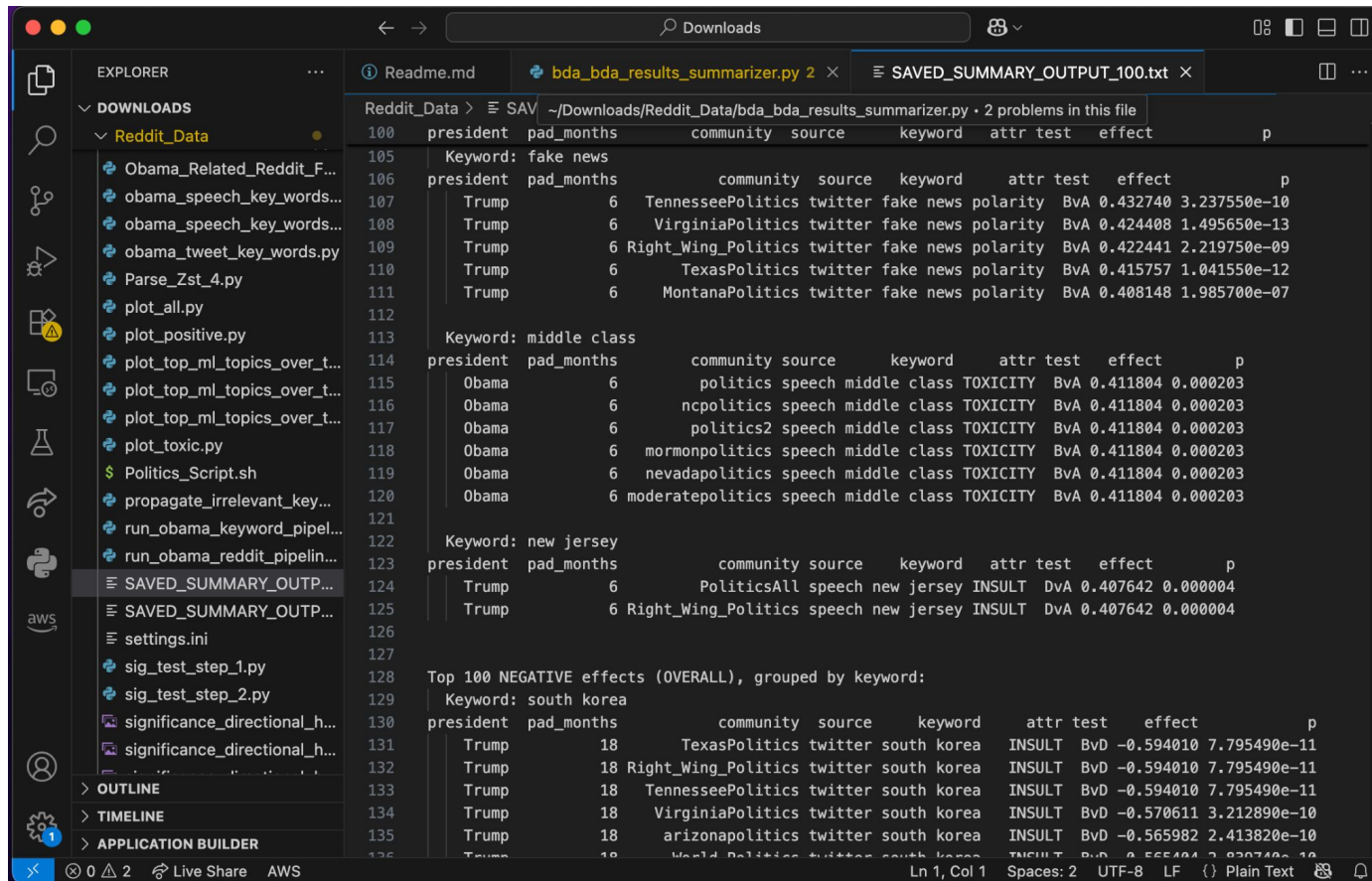
Program is an application on disk/flash memory
(static)



Process is a program when loaded into main memory (active)



Example of a Process



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `Obama_Related_Reddit_F...`, `obama_speech_key_words...`, `obama_tweet_key_words.py`, `Parse_Zst_4.py`, `plot_all.py`, `plot_positive.py`, `plot_top_ml_topics_over_t...`, `plot_top_ml_topics_over_t...`, `plot_toxic.py`, `Politics_Script.sh`, `propagate_irrelevant_key...`, `run_obama_keyword_pipel...`, `run_obama_reddit_pipel...`, `SAVED_SUMMARY_OUTP...`, `SAVED_SUMMARY_OUTP...`, `settings.ini`, `sig_test_step_1.py`, `sig_test_step_2.py`, `significance_directional_h...`, and `significance_directional_h...`. The code editor shows the output of a script, with the following content:

```
Readme.md bda_bda_results_summarizer.py 2 x SAVED_SUMMARY_OUTPUT_100.txt x
Reddit_Data > SAV ~/Downloads/Reddit_Data/bda_bda_results_summarizer.py · 2 problems in this file
100 president pad_months community source keyword attr test effect p
105 Keyword: fake news
106 president pad_months community source keyword attr test effect p
107 Trump 6 TennesseePolitics twitter fake news polarity BvA 0.432740 3.237550e-10
108 Trump 6 VirginiaPolitics twitter fake news polarity BvA 0.424408 1.495650e-13
109 Trump 6 Right_Wing_Politics twitter fake news polarity BvA 0.422441 2.219750e-09
110 Trump 6 TexasPolitics twitter fake news polarity BvA 0.415757 1.041550e-12
111 Trump 6 MontanaPolitics twitter fake news polarity BvA 0.408148 1.985700e-07
112
113 Keyword: middle class
114 president pad_months community source keyword attr test effect p
115 Obama 6 politics speech middle class TOXICITY BvA 0.411804 0.000203
116 Obama 6 ncpolitics speech middle class TOXICITY BvA 0.411804 0.000203
117 Obama 6 politics2 speech middle class TOXICITY BvA 0.411804 0.000203
118 Obama 6 mormonpolitics speech middle class TOXICITY BvA 0.411804 0.000203
119 Obama 6 nevadapolitics speech middle class TOXICITY BvA 0.411804 0.000203
120 Obama 6 moderatopolitics speech middle class TOXICITY BvA 0.411804 0.000203
121
122 Keyword: new jersey
123 president pad_months community source keyword attr test effect p
124 Trump 6 PoliticsAll speech new jersey INSULT DvA 0.407642 0.000004
125 Trump 6 Right_Wing_Politics speech new jersey INSULT DvA 0.407642 0.000004
126
127
128 Top 100 NEGATIVE effects (OVERALL), grouped by keyword:
129 Keyword: south korea
130 president pad_months community source keyword attr test effect p
131 Trump 18 TexasPolitics twitter south korea INSULT BvD -0.594010 7.795490e-11
132 Trump 18 Right_Wing_Politics twitter south korea INSULT BvD -0.594010 7.795490e-11
133 Trump 18 TennesseePolitics twitter south korea INSULT BvD -0.594010 7.795490e-11
134 Trump 18 VirginiaPolitics twitter south korea INSULT BvD -0.570611 3.212890e-10
135 Trump 18 arizonapolitics twitter south korea INSULT BvD -0.565982 2.413820e-10
136 Trump 18 WorldPolitics twitter south korea INSULT BvD -0.555404 2.830740e-10
```



How does the OS track Processes?

- Program Counter
- CPU Registers
- Stack Pointer

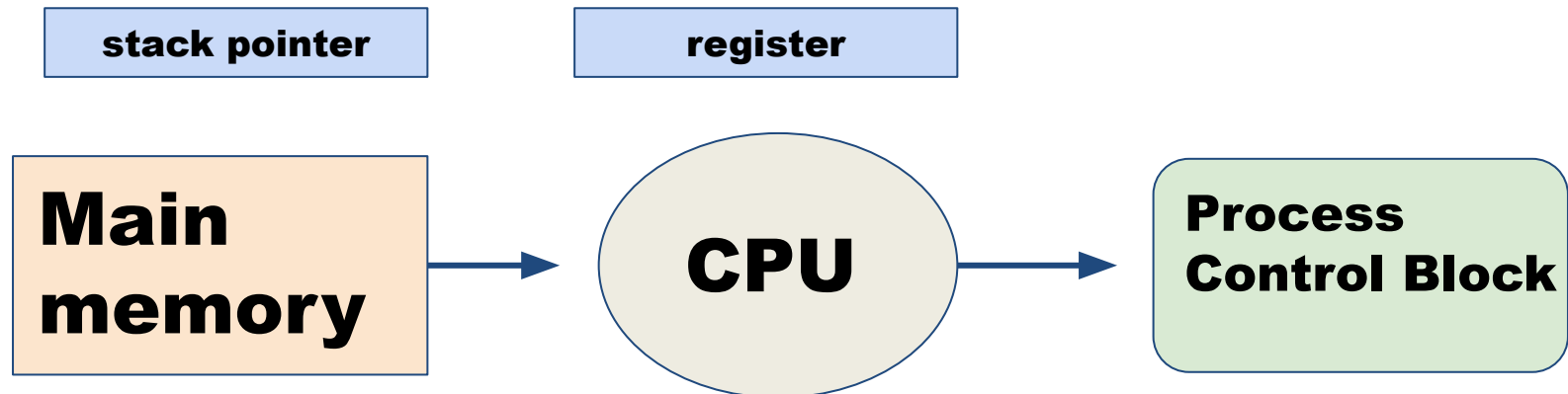
pc

```
sum = 0;
for (int i = 0; i < 10; ++i) {
    sum += i;
}
```

(a) Simple Loop Code

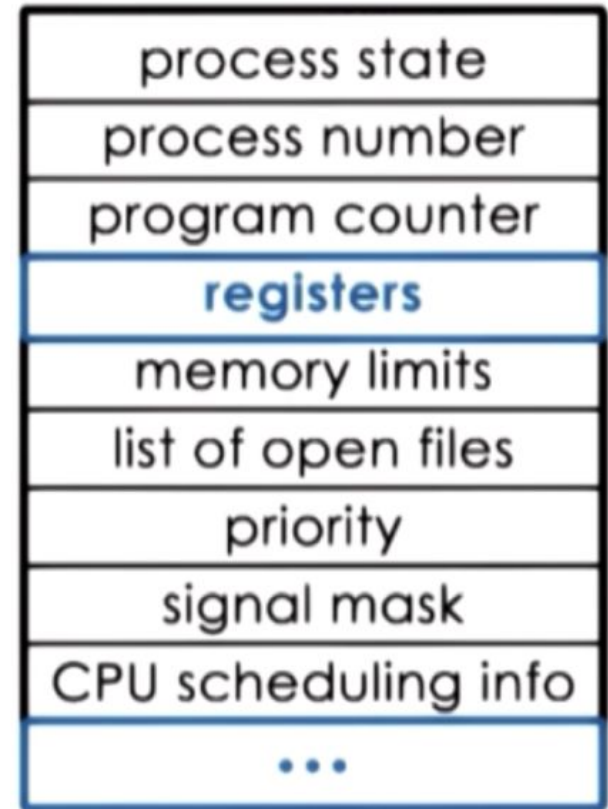
```
4004b8:  movl    $0x0,-0x8(%rbp)
4004bf:  movl    $0x0,-0x4(%rbp)
4004c6:  movl    $0x0,-0x4(%rbp)
4004cd:  jmp     4004d9 <main+0x25>
4004cf:  mov     -0x4(%rbp),%cax
4004d2:  add     %cax,-0x8(%rbp)
4004d5:  addl    $0x1,-0x4(%rbp)
4004d9:  cmpl    $0x9,-0x4(%rbp)
4004dd:  jle     4004cf <main+0x1b>
```

(b) Assembly Code

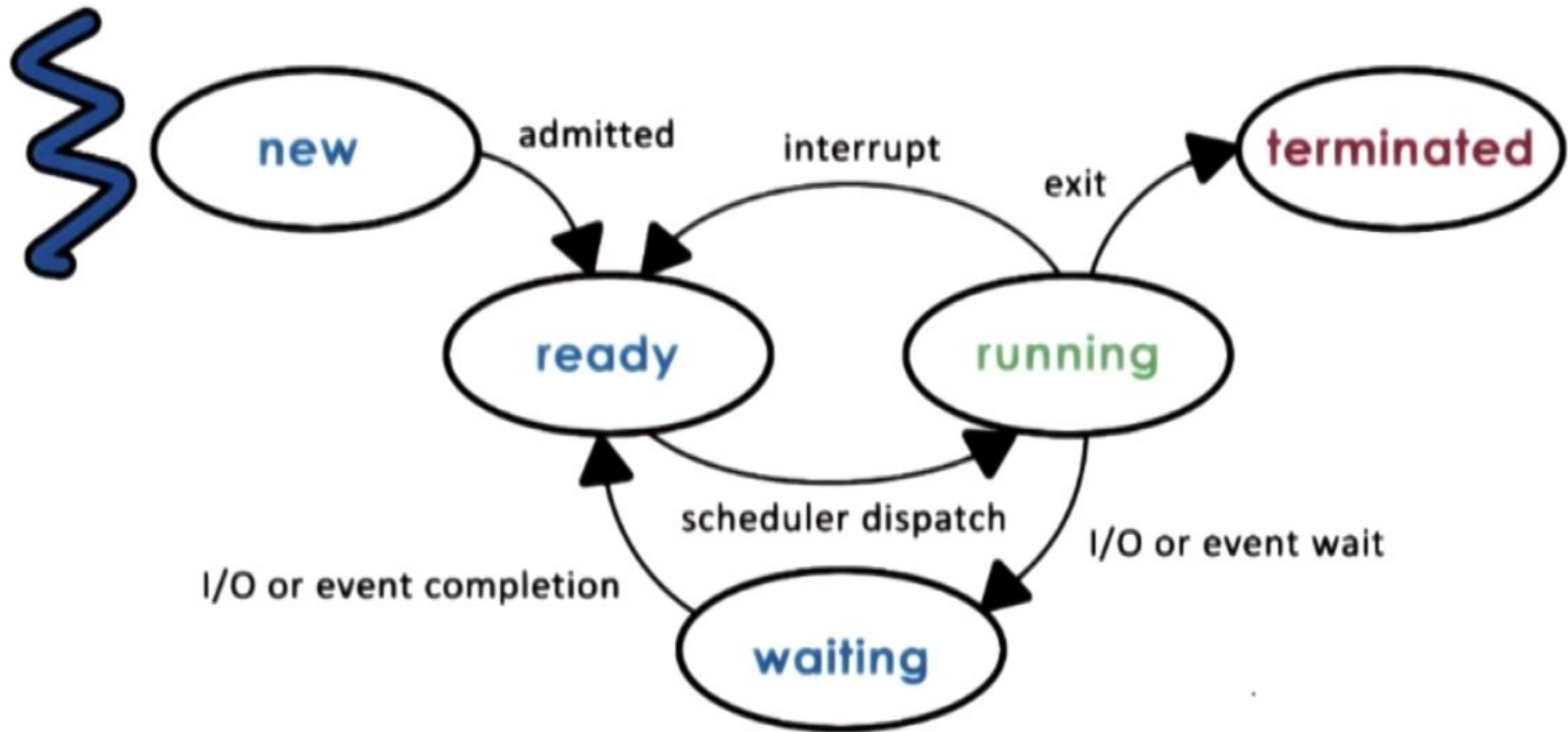


Process Control Block

- PCB created when process is created
- Certain fields are updated when process state changes
- Other fields change very frequently



Process Lifecycle: States



Process Lifecycle: States

- **New:** The process is being created
- **Running:** Instructions are being executed
- **Waiting:** The process is waiting for some event to occur (such as an I/O completion or reception of a signal)
- **Ready:** The process is waiting to be assigned to a processor
- **Terminated:** The process has finished execution



Processes

- Have an ID associated with it call the process ID (PID)

- **ps** command
 - PID: process ID
 - TTY: controlling terminal associated with the process
 - STAT: process status code
 - TIME: total CPU usage
 - CMD: name of executable/command

```
$ ps
```

PID	TTY	STAT	TIME	CMD
41230	pts/4	Ss	00:00:00	bash
51224	pts/4	R+	00:00:00	ps



/proc directory

- Contains virtual files and numbered directories corresponding to each running process
- Directory name = process ids
- When a process ends, its directory in /proc disappears automatically
- File in a number directory:
 - cmdline
 - cwd
 - environ
 - fd
 - maps, statm, mem, stat, status



/proc directory

- Some typical virtual files that provide
 - Hardware information:
 - `/proc/cpuinfo`: identifies the type of processor used by your system
 - `/proc/iomem`: shows you the current map of the system's memory for each physical device
 - `/proc/meminfo`
 - `/proc/interrupts`
 - File info
 - `/proc/filesystems`: displays a list of the file system types currently supported by the kernel
 - `/proc/partitions`
 - Kernel info
 - `/proc/cmdline`: shows the parameters passed to the kernel at the time it is started
 - `/proc/sys`



top command

- Displays a list of processes with their resource usage

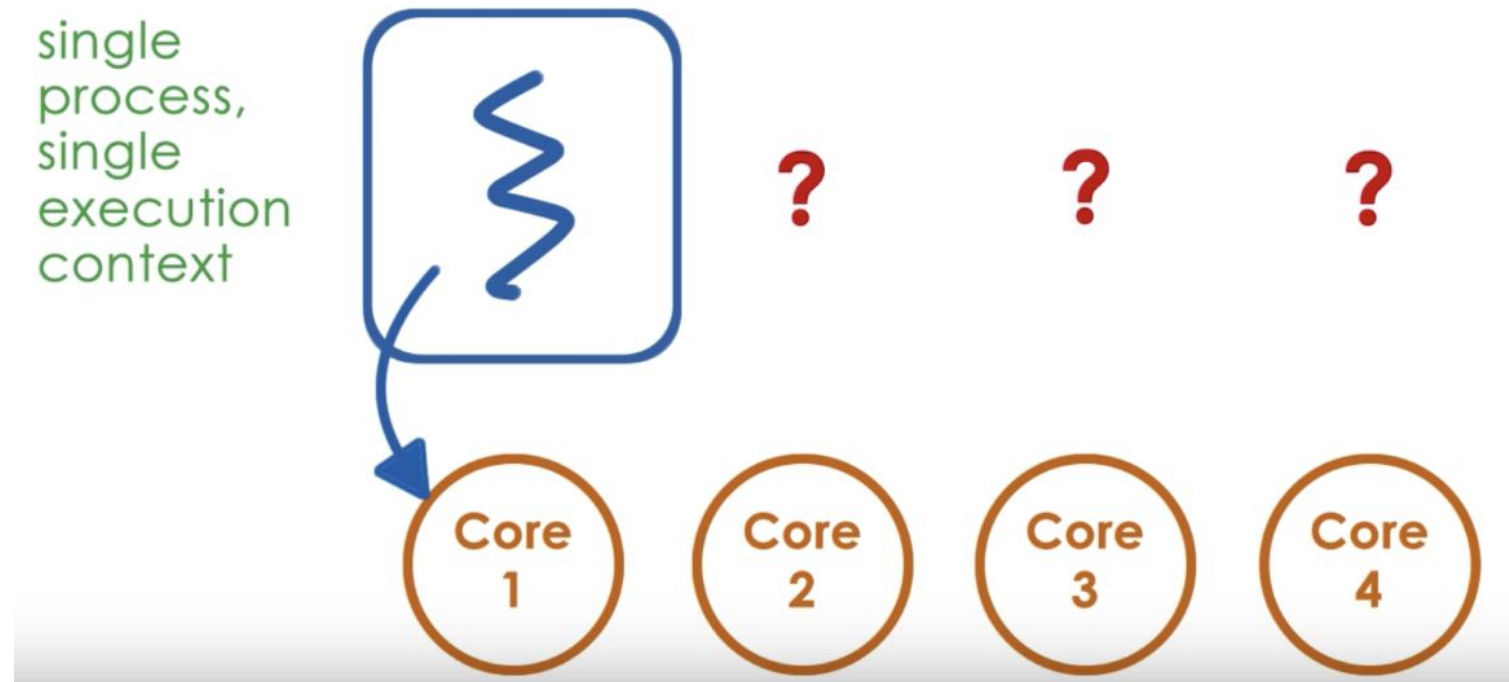
```
Terminal - user@cu-cs-vm: ~
File Edit View Terminal Tabs Help

top - 01:10:46 up 36 min, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 208 total, 1 running, 142 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.2 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2017320 total, 648132 free, 362208 used, 1006980 buff/cache
KiB Swap: 998396 total, 998396 free, 0 used. 1419228 avail Mem

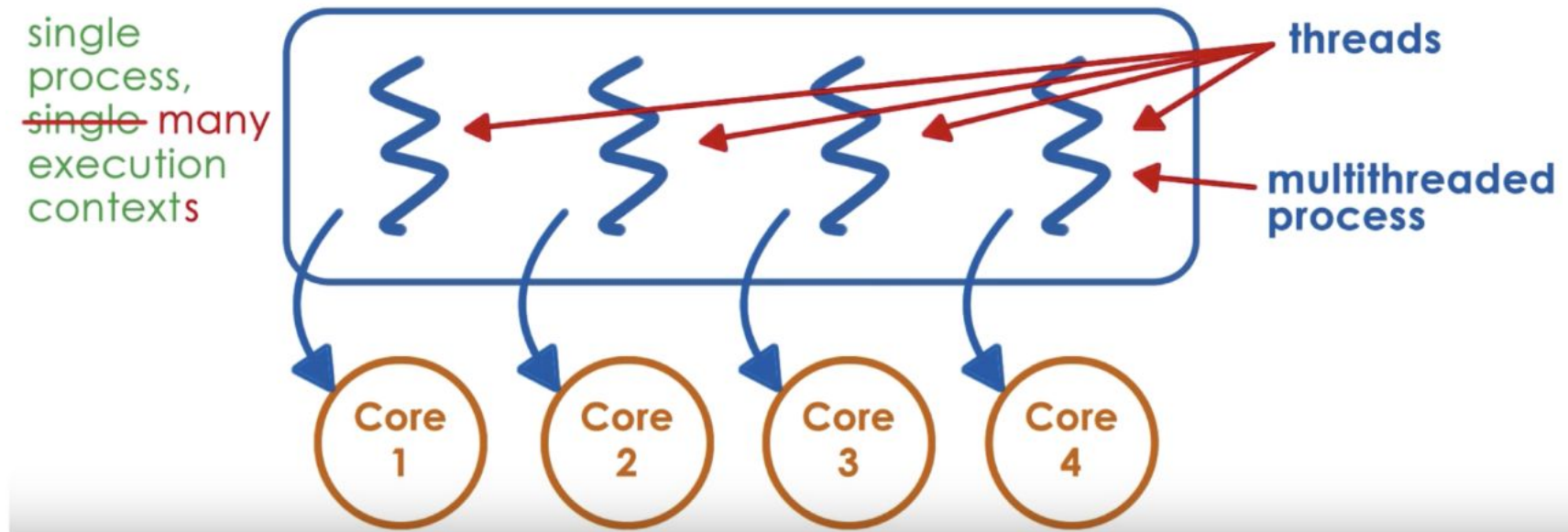
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 1362 user        20   0   439920   35304   28776 S   0.3   1.8   0:02.20 vmttoolsd
 2333 user        20   0    49288    3736    3096 R   0.3   0.2   0:00.09 top
    1 root         20   0   119996    6136    4020 S   0.0   0.3   0:03.12 systemd
    2 root         20   0         0         0         0 S   0.0   0.0   0:00.01 kthreadd
    3 root         20   0         0         0         0 I   0.0   0.0   0:00.06 kworker/0+
    4 root          0 -20         0         0         0 I   0.0   0.0   0:00.00 kworker/0+
    6 root          0 -20         0         0         0 I   0.0   0.0   0:00.00 mm_percpu+
    7 root         20   0         0         0         0 S   0.0   0.0   0:00.02 ksoftirqd+
    8 root         20   0         0         0         0 I   0.0   0.0   0:00.09 rcu_sched
    9 root         20   0         0         0         0 I   0.0   0.0   0:00.00 rcu_bh
   10 root         rt    0         0         0         0 S   0.0   0.0   0:00.00 migration+
   11 root         rt    0         0         0         0 S   0.0   0.0   0:00.00 watchdog/0
   12 root         20   0         0         0         0 S   0.0   0.0   0:00.00 cpuhp/0
   13 root         20   0         0         0         0 S   0.0   0.0   0:00.00 cpuhp/1
   14 root         rt    0         0         0         0 S   0.0   0.0   0:00.00 watchdog/1
   15 root         rt    0         0         0         0 S   0.0   0.0   0:00.00 migration+
   16 root         20   0         0         0         0 S   0.0   0.0   0:00.05 ksoftirqd+
```



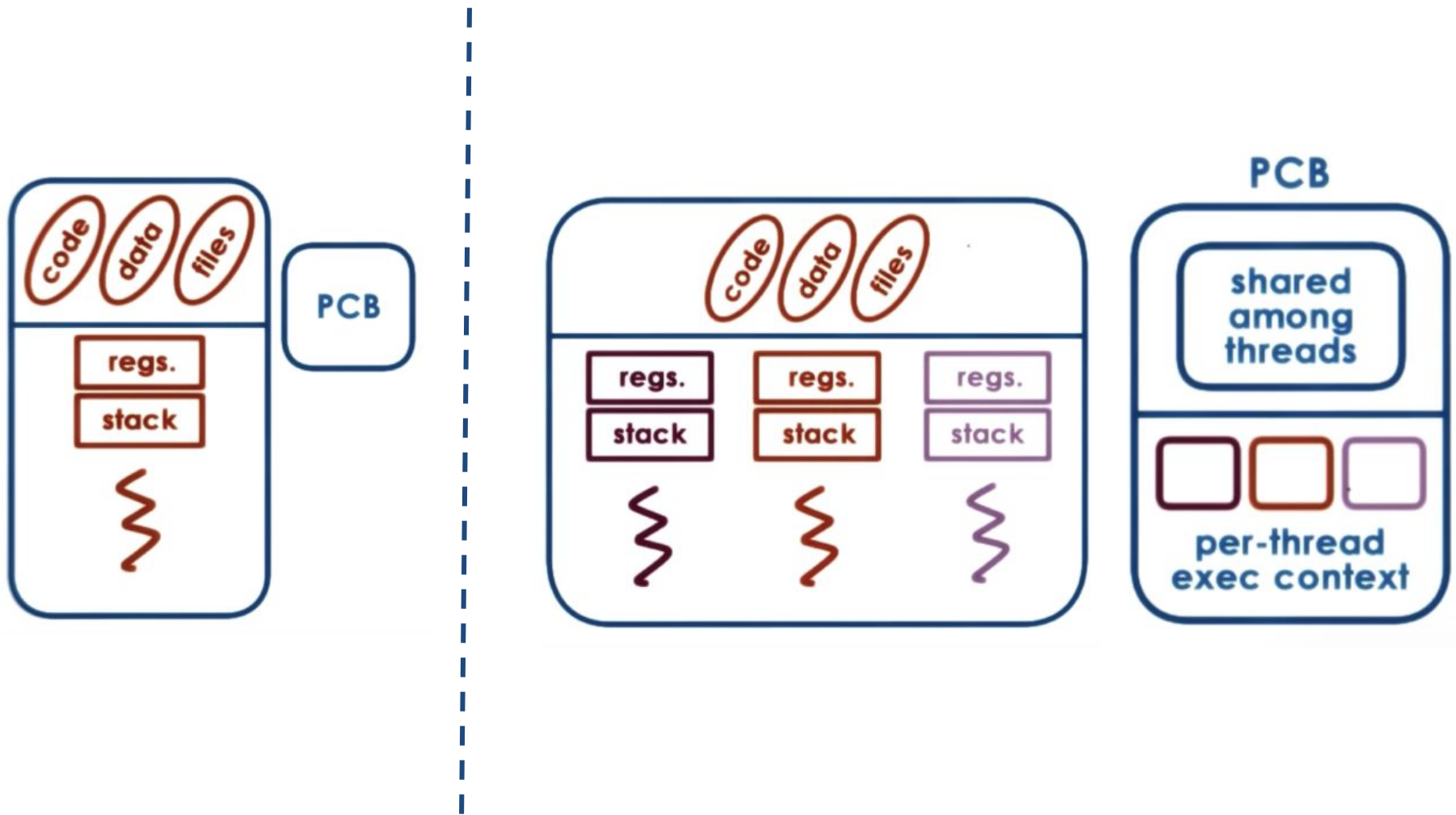
What if we have multiple CPUs?



What if we have multiple CPUs?



Process vs Threads



Process vs Thread

Process	Thread
Executing instance of an application	Path of execution within a process
Used for heavy weight tasks	Used for small tasks
Takes more time for termination and creation and context switching	Takes less time for termination and creation and context switching
Process consume more resources.	Threads consume less resources
Process is isolated	Threads share memory



Threads

- Threads share the same address space:
 - Code
 - Data
 - Other OS resources
 - heap segments
 - open file descriptors
 - signal and signal handlers
 - current working directory
 - user and group id
- Threads each have their own unique:
 - Thread ID
 - Stack
 - Set of registers
 - Program counter



Common Pitfalls with Multithreading

- Race Conditions
- Thread Safe Code
- Mutex deadlock



Common Pitfalls with Multithreading

- Race Conditions
 - While code may appear on the screen in the order you wish the code to execute, threads are scheduled by the operating system and are executed at random
- Thread Safe Code
- Mutex deadlock



Common Pitfalls with Multithreading

- Race Conditions
- Thread Safe Code
 - There should be no static or global variables which other threads may clobber or read assuming single threaded operation.
 - Many non-reentrant functions return a pointer to static data
→ Thread unsafe functions
- Mutex deadlock



Common Pitfalls with Multithreading

- Race Conditions
- Thread Safe Code
- Mutex deadlock
 - Occur when a mutex is applied but then not “unlocked”
 - Cause program execution to halt indefinitely



Threads Demo

- <https://tinyurl.com/CSCI3753>

